

Лабораторная работа 6

Вариант 34

ЗАДАНИЕ 1. ЗНАКОМСТВО СО СРЕДОЙ ПРОГРАММИРОВАНИЯ XCODE

1.1 УСТАНОВКА СРЕДЫ ПРОГРАММИРОВАНИЯ XCODE

Среда Xcode уже установлена

1.2 ЗАПУСК СРЕДЫ ПРОГРАММИРОВАНИЯ XCODE

Запустил Xcode



Все задания выполнил, с Xcode ознакомился

Вопросы:

1. Какие способы запуска симулятора iOS возможны?

либо сочетание клавиш `cmd+R`, или с помощью кнопки сборки и запуска проекта.

2. Как управлять симулятором?

С помощью инструмента `SimCtl`

ЗАДАНИЕ 2. КОНСОЛЬНОЕ ПРИЛОЖЕНИЕ НА OBJECTIVE-C

1) Изучить учебные материалы по созданию консольных приложений на языке Objective-C в Xcode

Все материалы изучил

2) Реализовать приложение на Objective-C согласно варианту.

Код:

```
#import <Foundation/Foundation.h>

int main(int argc, const char * argv[]) {
    @autoreleasepool {
        for (int i = 0; i < 1000; i++){
            if ((i / 100) % 2 != 0){
                int b = i % 10;
                int x = i % 100;
                int c = x / 10;
                int a = i / 100;
                if (a==b | a==c | b==c){
                    NSLog(@"%i%i%i", a, c, b);
                }
            }
        }
    }
    return 0;
}
```

3) Опубликовать код приложения в git-репозиторий в каталоге task2 и в ветке feature-task2.

The screenshot shows the GitHub web interface for a repository named 'feature-task2'. At the top, it indicates '2 branches' and '0 tags'. A message states 'This branch is 1 commit behind main.' Below this, a commit history table is visible:

Commit Message	Commit Hash	Time Ago	Commits
Alex and Alex task 2 completed	f31f011	3 minutes ago	2 commits

Below the commit history, a file list is shown:

File Name	Commit Message	Time Ago
task1	task 2 completed	3 minutes ago
.DS_Store	task 2 completed	3 minutes ago
README.md	readme	3 hours ago

Случайно назвал task1...

4) Предоставить протоколы тестов.

```
2022-04-15 12:39:35.192376-0700 task1[1882:49529] 755
2022-04-15 12:39:35.193189-0700 task1[1882:49529] 757
2022-04-15 12:39:35.193965-0700 task1[1882:49529] 766
2022-04-15 12:39:35.194876-0700 task1[1882:49529] 767
2022-04-15 12:39:35.195363-0700 task1[1882:49529] 770
2022-04-15 12:39:35.195698-0700 task1[1882:49529] 771
2022-04-15 12:39:35.196164-0700 task1[1882:49529] 772
2022-04-15 12:39:35.196743-0700 task1[1882:49529] 773
2022-04-15 12:39:35.197315-0700 task1[1882:49529] 774
2022-04-15 12:39:35.197918-0700 task1[1882:49529] 775
2022-04-15 12:39:35.198451-0700 task1[1882:49529] 776
2022-04-15 12:39:35.199157-0700 task1[1882:49529] 777
2022-04-15 12:39:35.200162-0700 task1[1882:49529] 778
2022-04-15 12:39:35.201157-0700 task1[1882:49529] 779
2022-04-15 12:39:35.202147-0700 task1[1882:49529] 787
2022-04-15 12:39:35.203137-0700 task1[1882:49529] 788
2022-04-15 12:39:35.204152-0700 task1[1882:49529] 797
2022-04-15 12:39:35.204707-0700 task1[1882:49529] 799
2022-04-15 12:39:35.205524-0700 task1[1882:49529] 900
2022-04-15 12:39:35.206441-0700 task1[1882:49529] 909
2022-04-15 12:39:35.207328-0700 task1[1882:49529] 911
2022-04-15 12:39:35.208007-0700 task1[1882:49529] 919
2022-04-15 12:39:35.209023-0700 task1[1882:49529] 922
2022-04-15 12:39:35.210006-0700 task1[1882:49529] 929
2022-04-15 12:39:35.210770-0700 task1[1882:49529] 933
2022-04-15 12:39:35.211102-0700 task1[1882:49529] 939
2022-04-15 12:39:35.211426-0700 task1[1882:49529] 944
2022-04-15 12:39:35.211754-0700 task1[1882:49529] 949
2022-04-15 12:39:35.212076-0700 task1[1882:49529] 955
2022-04-15 12:39:35.212397-0700 task1[1882:49529] 959
2022-04-15 12:39:35.213291-0700 task1[1882:49529] 966
2022-04-15 12:39:35.215109-0700 task1[1882:49529] 969
2022-04-15 12:39:35.216056-0700 task1[1882:49529] 977
2022-04-15 12:39:35.216410-0700 task1[1882:49529] 979
2022-04-15 12:39:35.216767-0700 task1[1882:49529] 988
2022-04-15 12:39:35.217089-0700 task1[1882:49529] 989
2022-04-15 12:39:35.217408-0700 task1[1882:49529] 990
2022-04-15 12:39:35.217758-0700 task1[1882:49529] 991
2022-04-15 12:39:35.218078-0700 task1[1882:49529] 992
2022-04-15 12:39:35.218396-0700 task1[1882:49529] 993
2022-04-15 12:39:35.218750-0700 task1[1882:49529] 994
2022-04-15 12:39:35.219075-0700 task1[1882:49529] 995
2022-04-15 12:39:35.219415-0700 task1[1882:49529] 996
2022-04-15 12:39:35.242794-0700 task1[1882:49529] 997
2022-04-15 12:39:35.243829-0700 task1[1882:49529] 998
2022-04-15 12:39:35.244756-0700 task1[1882:49529] 999
Program ended with exit code: 0
```

5) Ответить на вопросы.

Вопросы

1. Какие расширения файлов у следующих типов файлов?

- C language source file

.c

- C++ language source file

.cpp

- Header file

.h

- Objective-C source file

.m

- Objective-C++ source file

.mm

- Object (compiled) file

.o

2. Способы вывода на экран в Objective-C

NSLog, NSError и производные

ЗАДАНИЕ 3. КОНСОЛЬНЫЕ ПРИЛОЖЕНИЯ НА SWIFT: СПОСОБЫ СОЗДАНИЯ

1. Изучите примеры

Примеры изучил

2. Разработать приложение на языке Swift согласно варианту в консоли Swift REPL и в Xcode Playground.

3. Продемонстрировать навыки работы с консолью Swift REPL, компиляцию приложений с помощью swiftc, создание и тестирование консольных приложений в playground.

В Xcode Playground и Swift REPL невозможно реализовать ввод с консоли Xcode Playground (+тесты):

```

1  //: A Cocoa based Playground to present user interface
2
3  import AppKit
4  import PlaygroundSupport
5  import Foundation
6
7  let nibFile = NSNib.Name("MyView")
8  var topLevelObjects : NSArray?
9
10 Bundle.main.loadNibNamed(nibFile, owner:nil, topLevelObjects:&topLevelObjects)
11 let views = (topLevelObjects as! Array<Any>).filter { $0 is NSView }
12
13 // Present the view in Playground
14 PlaygroundPage.current.liveView = views[0] as! NSView
15
16 var x : Int
17 x = 1
18 var y : Int
19 y = 1
20
21 //1=y^2 + (x-1)^2
22 //(x - x0)^2 + (y - y0)^2 <= R^2
23 var a : Int
24 a = (x - 1) * (x - 1)
25 if (a + y * y <= 1) {
26     print("fits")
27 } else {
28     print("doesn't fit")
29 }

```



fits

```

16 var x : Int
17 x = 2
18 var y : Int
19 y = 1
20 |
21 //1=y^2 + (x-1)^2
22 //(x - x0)^2 + (y - y0)^2 <= R^2
23 var a : Int
24 a = (x - 1) * (x - 1)
25 if (a + y * y <= 1) {
26     print("fits")
27 } else {
28     print("doesn't fit")
29 }

```



doesn't fit

```
16 var x : Int
17 x = 10
18 var y : Int
19 y = 11
20
21 //1=y^2 + (x-1)^2
22 //(x - x0)^2 + (y - y0)^2 <= R^2
23 var a : Int
24 a = (x - 1) * (x - 1)
25 if (a + y * y <= 1) {
26     print("fits")
27 } else {
28     print("doesn't fit")
29 }
```

doesn't fit

Swift REPL:

```
[alexmac@Macs-iMac ~ % swift
Welcome to Apple Swift version 5.1.3 (swiftlang-1100.0.282.1 clang-1100.0.33.15).
Type :help for assistance.
[ 1> var x : Int
x: Int = 0
[ 2> x = 10
[ 3> var y : Int
y: Int = 0
[ 4> y = 5
[ 5> var a : Int
a: Int = 0
[ 6> a = (x-1)*(x-1)
[ 7> if (a + y*y <= 1){
[ 8.     print("fits")
[ 9. } else {
[10.     print("doenst fit")
[11. }
doenst fit
```

Swiftc:

```
[alexmac@Macs-iMac labrabota6-gr15-alekseykrazhev % swiftc task3.swift
[alexmac@Macs-iMac labrabota6-gr15-alekseykrazhev % ./task3
doesn't fit
alexmac@Macs-iMac labrabota6-gr15-alekseykrazhev %
```

Задания 5-6: скриншоты выше.

7. Ответить на вопросы.

Вопросы

1. Что такое swift REPL и в каких случаях рекомендуется использовать?

swift REPL позволяет выполнять код в режиме реального времени и сразу видеть результат. Удобно для дебага небольших проектов.

2. Что такое playground? Какие задачи (проекты) можно создавать в playground?

Playground - дословно “песочница”, позволяет легко изучить swift. Вы пишете строчку кода и видите результат ее выполнения.

С его помощью можно просмотреть каждый шаг алгоритма, выучить синтаксис swift, изучить новые API.

3. Как компилировать приложение на языке swift в консоли bash?

```
swiftc -o <filename> <filename>.swift или swiftc <filename>.swift
```

```
./<filename>
```

ЗАДАНИЕ 4. ПОДКЛЮЧЕНИЕ РЕПОЗИТОРИЯ И ОРГАНИЗАЦИЯ РАБОТЫ С ВЕТКАМИ В XCODE

Все примеры изучил

ВОПРОСЫ

1. Как создать локальный git-репозиторий для проектов в Xcode?

При запуске Xcode выбрать пункт “Создать репозиторий”

3. Как добавить внешний репозиторий в Xcode? Назвать два способа подключения.

```
source control -> clone
```

2. Как создать ветку репозитория в Xcode? 3. Как отменить commit в Xcode?

```
branch from main
```

```
git hist -> список всех коммитов
```

```
найти нужный -> копировать 7 символов хэша
```

```
Далее git reset --hard имя_коммита_перед_неверным
```

```
-> мы сбросили ветку до последнего верного коммита.
```

4. Как слить ветки в Xcode?

```
merge into branch
```

ЗАДАНИЕ 5. КОНСОЛЬНОЕ ПРИЛОЖЕНИЕ ДЛЯ ОБРАБОТКИ ТЕКСТОВЫХ ДАННЫХ

Задания 1-5 выполнил

6. Предоставить протоколы тестов.

```
Amount of data = 4
Evroopt: Optional("150")
All elements in Shop:
Evroopt: 150
Zara: 20
Angelo: 10
Perekrestok: 50
Fish: Optional("2022")
Zara: Optional(25)
Shop Dict sorted by key:
Angelo: 10
Evroopt: 150
Perekrestok: 50
Zara: 25
Shop Dict sorted by key:
Angelo: 10
Zara: 25
Perekrestok: 50
Evroopt: 150
Removed Perekrestok from Shop
Zara: 25
Angelo: 10
Evroopt: 150
Deleted all elements from Product successfully
Program ended with exit code: 0
```

7. Включить скриншоты и код приложений в отчет.

Код:


```

import Foundation

// initialize empty Dictionary
var Product : Dictionary<String, Int> = [:]
// initialize Dictionary with data
var Shop: Dictionary<String, Int> = ["Perekrestok": 50, "Evroopt": 150, "Zara": 20, "Angelo": 10]

// count amount of data in Dictionary
print("Amount of data = ", Shop.count)

// print element by key
var key: String = "Evroopt"
print("\(key): \(String(describing: Shop[key]?.description))")

// print all elements in Dictionary
print("All elements in Shop:")
for (shop, val) in Shop{
    print ("\(shop): \(val) ")
}

// Add element to Dictionary
Product["Fish"] = 2022
print("Fish: \(String(describing: Product["Fish"]?.description))")

// modify elements in Dictionary
Shop.updateValue(25, forKey: "Zara")
print("Zara: \(String(describing: Shop["Zara"]))")

// sort by key
let sorted = Shop.sorted { $0.key < $1.key }
print("Shop Dict sorted by key:")
for (shop, val) in sorted{
    print ("\(shop): \(val) ")
}

// sort by element
let sorted1 = Shop.sorted { $0.value < $1.value }
print("Shop Dict sorted by key:")
for (shop, val) in sorted1{
    print ("\(shop): \(val) ")
}

// delete element
Shop.removeValue(forKey: "Perekrestok")
print("Removed Perekrestok from Shop")
for (shop, val) in Shop{
    print ("\(shop): \(val)")
}

// delete all elements
Product.removeAll()
if (Product.isEmpty) {
    print("Deleted all elements from Product successfully")
} else {
    print("Delete failed")
}

```

ВОПРОСЫ

1) Расскажите о режимах работы среды программирования xCode.

Runtime Issues оповещает о дефектах, которые автоматически обнаруживает Xcode. Thread Sanitizer отслеживает изменение данных и прочие баги. Проверку интерфейса осуществляет View Debugger — обновляющийся инструмент с высокой визуальной точностью. Memory Debugger оповещает об «утечках памяти» и скрытых багах.

2) В чем отличия языка программирования Swift от C и C++.

Есть именованные параметры, необязательно ставить “:” , отсутствуют указатели -> повышена безопасность.

3) Какие способы отладки программ поддерживает среда xCode?

расставить breakpoint (точки останова) и посмотреть результаты действий

консольный отладчик LLDB.

4) Какие операторы языка swift Вы знаете?

операторы сравнения, логические операторы, операторы арифметических действий, бинарные, унарные, тернарные (условный оператор: выражение ? действие1 : действие2)

5) Какие типы коллекций применяются в языке Swift и в чем их особенности?

Массивы (массивы хранят много значений одинакового типа в упорядоченном списке. Одно и то же значение в массиве может появиться несколько раз, в разных позициях.),

множества (множество хранит различные значения одного типа в виде коллекции в неупорядоченной форме)

и словари (словарь представляет собой контейнер, который хранит несколько значений одного и того же типа. Каждое значение связано с уникальным ключом, который выступает в качестве идентификатора этого значения внутри словаря)

6) Какие типы проектов xCode Вы знаете?

Single View Application - шаблон, в котором готово одно окно

Empty Application - пустой проект

Tabbed Application - шаблон для проекта с несколькими окнами

OpenGL Game - для написания игр на основе OpenGL.